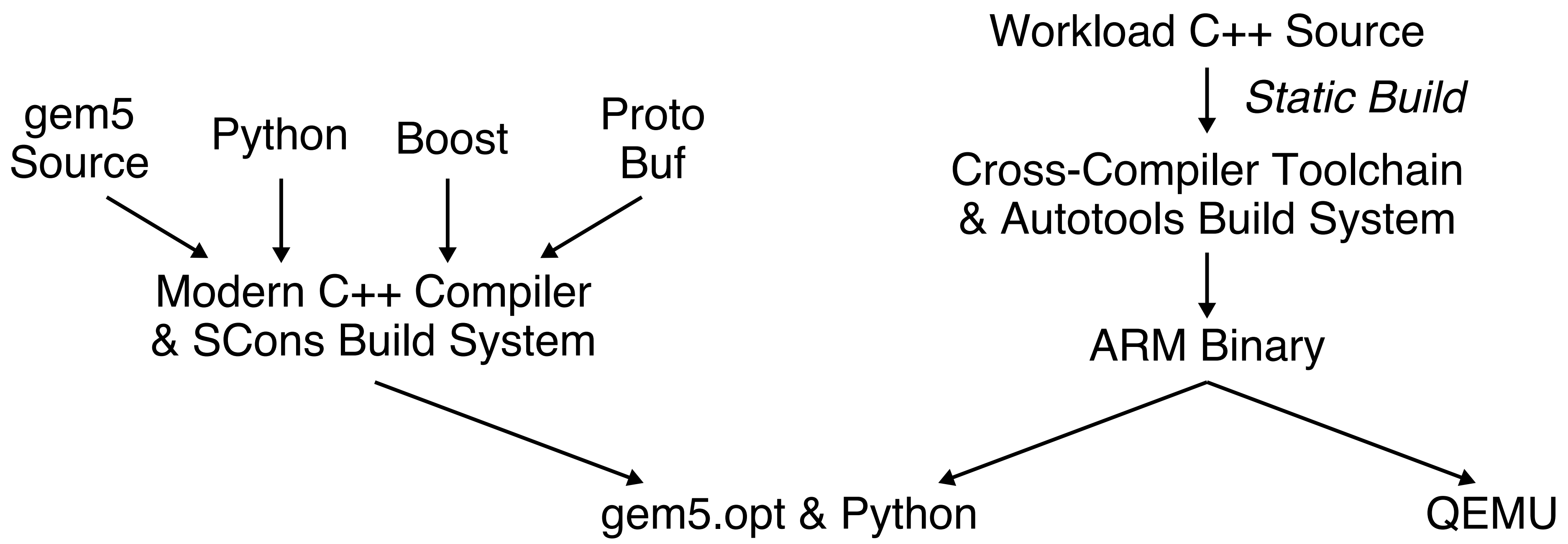
The Case for Using Guix to Solve the gem5 Packaging Problem

Christopher Batten[1], Pjotr Prins[2]
Efraim Flashner[2], Arun Isaac[2], Ekaiz Zarraga[3]
Erik Garrison[2], Tuan Ta[1]

[1] School of Electrical and Computer Engineering, Cornell University
[2] The University of Tennessee Health Science Center
[3] ElenQ Technology

The gem5 Packaging Problem



- **gem5 Simulator Packaging**
  - Numerous build/run-time deps
  - Numerous build-time options (diffferent ISAs, coherence protocols, accelerators)
  - Everyone builds from scratch
- **gem5 Workload Packaging**
  - Build cross-compiler toolchain
  - Build an emulator for testing
  - Possibly ensure static linking
  - Everyone duplicates this work

Currently no gem5 packages!

An Ideal gem5 Packaging Solution?

- **Reproducible –** deterministically duplicate development environment
- **Transparent –** understand entire development environment including exact build configurations and version of every dependency
- **Lightweight –** integrate into standard development environment
- **Flexible –** easily switch between different development environments
- **Portable –** build gem5 workloads for native execution and/or target multiple ISAs for cycle-level simulation
- **Fast –** leverage precompiled packages when available
- **Distribution Agnostic –** enable researchers to use any distribution
- **Extensible –** extensions through a general-purpose language

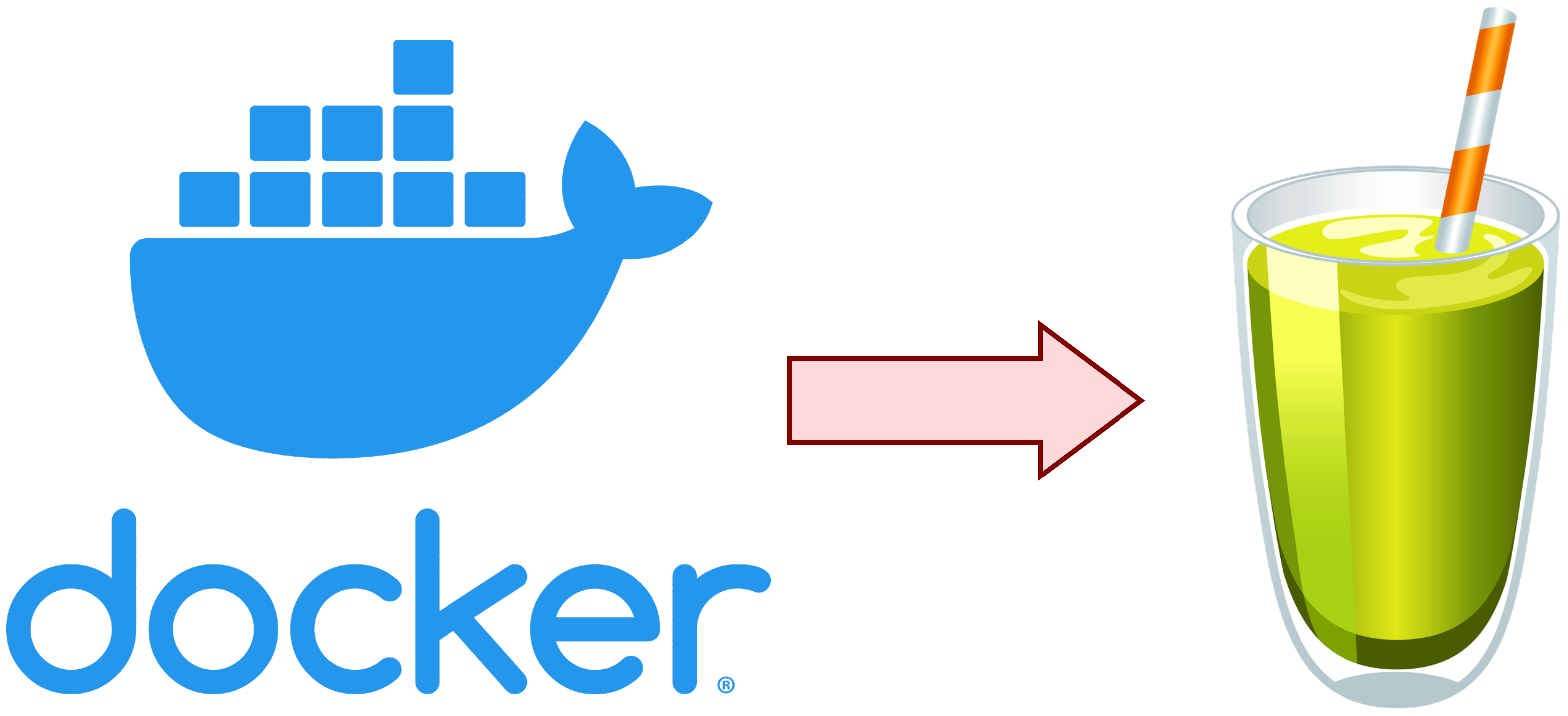
Why not just use Docker?

```
FROM ubuntu:20.04
ENV DEBIAN_FRONTEND=noninteractive
RUN apt -y update
RUN apt -y upgrade
RUN apt -y install build-essential git m4 scons zlib1g zlib1g-dev
                   libprotobuf-dev protobuf-compiler libprotoc-dev ...
RUN pip install mypy
```

```
% git clone https://gem5.googlesource.com/public/gem5
% docker pull gcr.io/gem5-test/ubuntu-20.04_all-dependencies:v21-2
% docker images
REPOSITORY                                      SIZE
gcr.io/gem5-test/ubuntu-20.04_all-dependencies  1.38GB
% docker run -u $UID:$GID --volume /home/cb535/gem5:/gem5
    --rm -it gcr.io/gem5-test/ubuntu-20.04_all-dependencies:v21-2
I have no name!@bbfd8a86240b:/$
```

- Reproducible?
- Transparent?
- Lightweight?
- Flexible?
- Portable?
- Fast?
- Distribution Agnostic?
- Extensible?

Containers are Like Smoothies



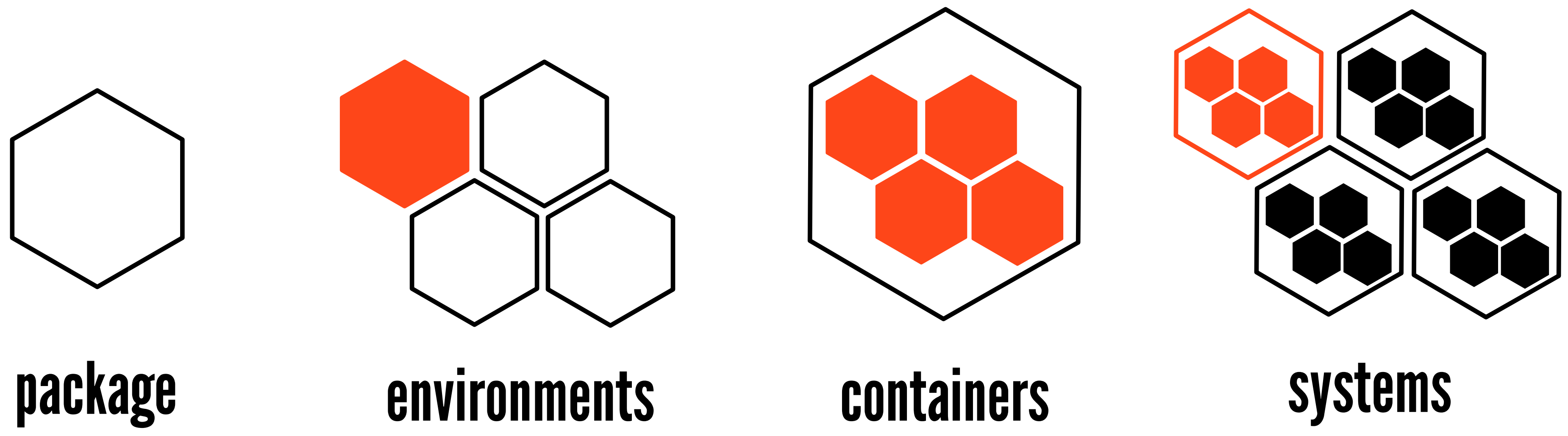
A smoothie tastes great ...
but how much do we know about what is really in the smoothie?
Can someone else make the exact same smoothie?

Adapted from L Courtès, FOSDEM'20

Talk Outline

What is Guix?

General toolbox for software deployment



- Guix is a functional, transactional package manager
- Guix is an environment manager
- Guix is a reproducible container generator
- Guix is a complete operating system constructor and manager

Adapted from L Courtès, FOSDEM’20

Guix Hello World

```
% guix pull
% guix install hello
% guix package --list-installed
hello 2.12 /gnu/store/x2byq2a04pi...1mqikz07i1m-hello-2.12

% which hello
~/.guix-profile/bin/hello

% readlnk $(which hello)
/gnu/store/x2byq2a04pi...1mqikz07i1m-hello-2.12/bin/hello

% hello
Hello, world!

% guix remove hello
```

Guix is more than a package manager!

The Guix `hello` Package

```
(define-public hello
(package
  (name "hello")
  (version "2.12.1")
  (source (origin
            (method url-fetch)
            (uri (string-append "mirror://gnu/hello/hello-" version
                                ".tar.gz"))
            (sha256
             (base32
              "086vqwk2wl8zfs47sq2xpjc9k066ilmb8z6dn0q6ymwjzlm196cd"))))
  (build-system gnu-build-system)
  (synopsis "Hello, GNU world: An example GNU package")
  (description
   "GNU Hello prints the message \"Hello, world!\" and then exits.  It
serves as an example of standard GNU coding practices.  As such, it
supports command-line arguments, multiple languages, and so on.")
  (home-page "https://www.gnu.org/software/hello/")
  (license gpl3+)))
```

Talk Outline

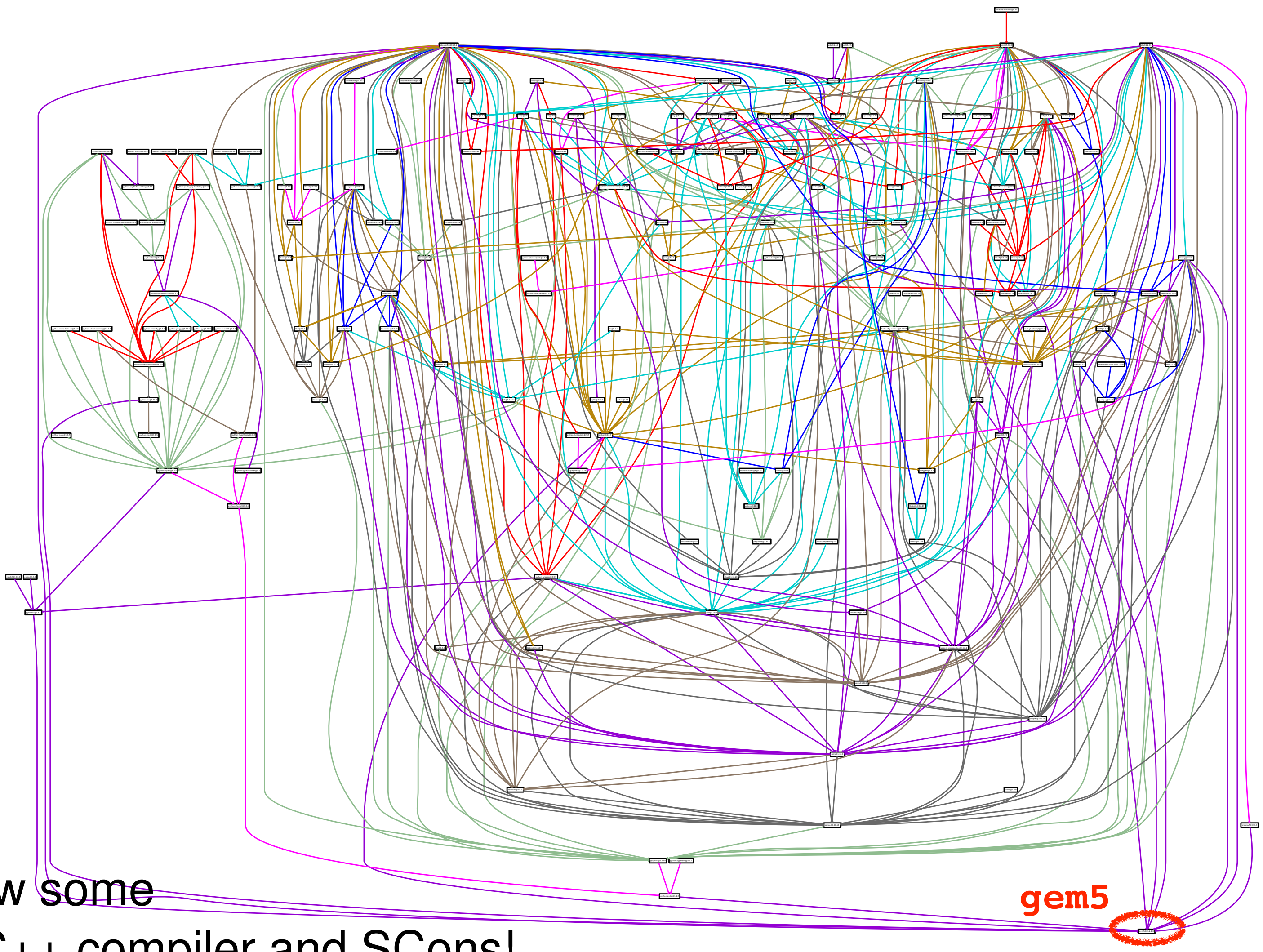
The Guix `gem5` Package

`https://git.genenetwork.org/guix-bioinformatics/guix-bioinformatics/src/branch/master/gn/packages/virtualization.scm#L21`

- Fetch specific version using git tag
- Eliminate non-deterministic use of `__DATE__` and `__TIME__`
- Patch `Makefile`/`SConstruct` to use Guix packages such as `pybind11`, `zlib`, `libpng`
- Leverage Guix built-in support for SCons build systems
- Builds for multiple architectures (e.g., x86, ARM, RISC-V)
- Installs binaries for each simulator suffixed with architecture
- Installs default configurations
- Captures all dependencies
- Provides a derived package to install a single architecture

The Guix `gem5` Package Dependency Graph

```
(inputs
 (list
   gperftools
   libpng
   protobuf
   pybind11
   python
   python-pydot
   python-six
   zlib))
(native-inputs
 (list
   boost
   m4
   pkg-config))
```



Does not even show some dependencies on C++ compiler and SCons!

The Guix gem5 Package Hash

```
% guix install gem5
% readlink $(which gem5-arm.opt)
/gnu/store/4n7fh47hlgzmmwj744j1qz9llc70kbmz-gem5-21.2.1.0
  /bin/gem5-arm.opt
```

This hash in the `/gnu/store` captures:

- all direct dependencies (e.g., gperftools, protobuf, pybind11, zlib, etc)
- all implicit dependencies (e.g., C++, SCons, etc)
- all recursive dependencies (e.g., pytest, readline, expat, etc)
- even the compiler used to build the compiler!
- every command line option and environment variable

gem5

&

Guix

Talk Outline

The Guix `smithwaterman` Package

```
(define-public smithwaterman-static
  (package
    (inherit smithwaterman)
    (name "smithwaterman-static")
    (arguments
     (substitute-keyword-arguments
       (package-arguments smithwaterman)
         ((#:make-flags flags ''())
          #~(cons "CFLAGS=-static" #$flags))))))

% guix build --target=aarch64-linux-gnu \
    smithwaterman-static
/gnu/store/4kq8lc9z50vgzlzgdavqgffxzvpbwpx3
            -smithwaterman-static-0.0.0-2.2610e25
```

Talk Outline

Guix for gem5 Demo

```
% guix install smithwaterman
% smithwaterman -p TGATTGTACCAAA TGATCATGTACCA

% guix install qemu gem5

% DIR=$(guix build \
         --target=aarch64-linux-gnu smithwaterman-static)
% ln -sf $DIR/bin/smithwaterman sw
% qemu-aarch64 ./sw -p TGATTGTACCAAA TGATCATGTACCA
% gem5-arm.opt \
    $GUIX_PROFILE/share/gem5/configs/example/se.py \
    --cmd=./sw \
    --options="-p TGATTGTACCAAA TGATCATGTACCA"
```

Guix for gem5 Demo

```
% gem5-arm.opt \
    --outdir=m5out-io-sw \
    $GUIX_PROFILE/share/gem5/configs/example/se.py \
    --cpu-type=MinorCPU --ruby --cmd=./sw \
    --options="-p TGATTGTACCAAA TGATCATGTACCA" \

% gem5-arm.opt \
    --outdir=m5out-o3-sw \
    $GUIX_PROFILE/share/gem5/configs/example/se.py \
    --cpu-type=O3CPU --ruby --cmd=./sw \
    --options="-p TGATTGTACCAAA TGATCATGTACCA" \

% grep system.cpu.numCycles m5out-io-sw/stats.txt
% grep system.cpu.numCycles m5out-o3-sw/stats.txt
```

gem5

&

Guix

Take-Away Points

- Packging the gem5 simulator and gem5 workloads can be challenging
- Guix is a mature toolbox for software deployment including support for packages, environments, containers, and systems
- Guix can potentially offer a compelling option for packaging the gem5 simulator and gem5 workloads